Global Optimization of Atomic and Molecular Clusters Using the Space-Fixed Modified Genetic Algorithm Method

J. A. NIESSE, HOWARD R. MAYNE

Department of Chemistry, University of New Hampshire, Durham, New Hampshire 03824

Received 6 September 1996; accepted 13 January 1996

ABSTRACT: A modified genetic algorithm approach has been applied to atomic Ar clusters and molecular water clusters up to $(H_2O)_{13}$. Several genetic operators are discussed which are suitable for real-valued space-fixed atomic coordinates and Euler angles. The performance of these operators has been systematically investigated. For atomic systems, it is found that a mix of operators containing a coordinate-averaging operator is optimal. For angular coordinates, the situation is less clear. It appears that inversion and two-point crossover operators are the best choice. © 1997 by John Wiley & Sons, Inc. *J Comput Chem* **18**: 1233–1244, 1997

Keywords: global minimization techniques; genetic algorithm; search techniques; clusters; cluster geometry

Introduction

The most stable structure of a molecular or an atomic or molecular cluster is often the geometry with the lowest potential energy. The location of this geometry is clearly of great interest to chemist.[1–3] Unfortunately, the task of minimizing the energy is a formidable one. This is due to the large number of minima which even small molecules and clusters can possess; the number of minima increases rapidly with the number of atoms in the system. For example, a cluster of 13 atoms bound by Lennard–Jones (LJ) interactions possesses on the order of 10^3 local minima.[4–7] In fact, it is estimated[6,7] that the number of local minima for $(LJ)_n$ grows as rapidly as $\exp(n^2)$. Clearly an exhaustive systematic search in all dimensions is not feasible. It is also extremely improbable that a random walk in the configuration space is likely to find the global minimum. Therefore, several strategies for guiding the search have been devised.

Among the commonly used techniques for locating extrema of functions are those, such as the conjugate gradient (CG) method,[8] which use the gradient of the potential energy surface to slide

Correspondence to: H. R. Mayne

CCC 0192-8651 / 97 / 091233-12

downhill. Such techniques typically locate the closest local minimum (in the sense that the minimum acts as an attractor, and all points within its basin are attracted to it). Such methods, which move to a nearby local minimum, are sometimes termed ''greedy''; they solve the local minimization problem at the expense of the global solution. Clearly, the greatest difficulty in locating the global minimum of the potential energy hypersurface is the presence of a vast number of local minima. Therefore, one goal of a minimization search is to efficiently escape from local minima. For the search to be successful, this must be to a minimum of lower potential energy.

There have been several recent attempts to introduce methods that allow escape from these local minima. The popular method of simulated annealing (SA)[8,9] has been successful in this arena. The technique is an extension of Metropolis Monte Carlo techniques,[10] in which the system point is allowed to wander over the configuration space at a given temperature. In SA, the temperature is initially high, then is slowly reduced to absolute zero. At the end of the cooling, if the cooling is done infinitely slowly, the system will have ''frozen'' at the global minimum. In practice, however, infinitely slow cooling is not realizable. The presence of high barriers at the saddle points between minima may, at low temperatures, make the transition between minima extremely slow. In other words, the system may be kinetically, rather than thermodynamically, controlled.

A related technique, called jump-walking Monte Carlo,[11] generates a canonical distribution by occasionally sampling higher temperatures. This technique has been used to overcome slow barrier crossing and is used as a minimization technique in $(LJ)_{13}$ and $(H_2O)_8$ clusters.[12]

A rather different strategy has been to allow the system to behave quantum mechanically, leading to the possibility of tunneling. This has been accomplished by using Gaussian wave packets in imaginary time,[13] by using distributed Gaussians,[14] and by combining simulated annealing and quantum Monte Carlo.[15] One problem with such methods is that they become increasingly difficult to implement as the number of dimensions increases.

Recently, there has been considerable interest in a stochastic global minimization technique which requires no such quantum formulation, and is undeterred by barriers. This is the method of genetic algorithms.[16–19] The genetic algorithm (GA) technique is inspired by concepts from Darwinian natural evolution. Populations of candidate solutions compete with each other for survival. Through selection, breeding, and mutation operations, the fittest individuals pass their genetic characteristics on to later generations. In this way, it is hoped that the ultimate surviving individual (in this case, the cluster geometry) is the fittest possible; that is, the best solution to the optimization problem posed (in this case, the global minimum).

One advantage of the GA method is that it is not ''greedy.'' The genetic operators often create children whose structures differ drastically from their parents. Thus, these operators allow the system to escape from local minima, because the system does not attempt to ''creep'' uphill. In this way, there can be extensive searching of the configuration space, with the search appropriately guided by the fitness function.

There have been several recent applications of the GA technique in the chemical literature dealing with the problem of obtaining the minimum energy geometry of a large molecule or cluster.[20–33] In such cases, the ''fitness'' of a geometry is some function of its potential energy, with low potentials having higher fitness. Several studies have applied the technique to conformational studies of organic molecules and for protein side-chains.[20–24] Several have modeled protein binding or solvation as a small cluster and have used GA in minimizing the potential energy of the cluster.[25,26] Other investigations have focused on small atomic[27–34] and molecular[34] clusters. In several of these works, GA and other minimization techniques are directly compared.

Many recent developments in the computational technology for GA applications have been made in the area of atomic cluster calculations. For instance, Zeiri[28] employed Cartesian variables and an array of simple operators to confirm published[38] geometries of $(Ar)_nH_2$ clusters. He found his calculations to be, at worst, comparable to SA. Deaven and Ho[29] were able to locate the global minimum for C_{60} using GA, although SA was unable to do so. Their calculations used spaced-fixed coordinates, and a novel genetic operator, together with a gradient-based local minimization of each offspring produced. In an impressive performance, the same group[30] was also able to locate the global minimum of $(LJ)_n$ for n up to 100 using the same method. Gregurick et al.[33] and the present investigators[34] have used a GA approach to minimize clusters as large as $n = 29$ and 55, respectively, and give detailed timing statistics. The present investigators[35] have also employed the technique

to minimize small silicon clusters on a semiempirical potential energy surface.

The most powerful of these techniques[29,30,33–35] incorporate an important innovation: they all use a gradient-based (conjugate gradient) local descent for each geometry generated. By introducing this measure the algorithm could avoid searching parts of the configuration space which are repulsive. Points on the potential energy surface which happen to be very high in energy (therefore unfit) can, in fact, be ''close'' (in the sense that they are moved there by a gradient-based descent) to geometries which have low potential energy (and are therefore fit).

The work of Gregurick et al.[33] differs slightly from that of Deaven, Ho, and coworkers.[29,30] and the present investigators.[34,35] The former group used a conjugate gradient descent on the offspring geometries only to compute the fitness of that offspring. They then continued the calculation with the originally produced geometry. The other two groups used the geometries of the local minima as the offspring. In essence, these workers restricted the individuals in their populations to be the geometries of local minima. Thus, the search has become a search through a finite (albeit large) number of individuals, rather than over an infinite set of possibilities.

Recently, it has become clear that the GA approach using real-valued Cartesian variables and appropriate operators is more powerful than was anticipated by traditional GA workers. The present investigators found that the method could outperform binary coded algorithms for LJ clusters both in terms of timing and in the size of the search space[34] Deaven, Ho, and coworkers[29,30] have managed to minimize LJ clusters up to $n = 100$ using GA. They point out that this is the first time any single technique has been used to minimize this entire range of structures. Furthermore, their calculations detected previously unreported minima in several cases. The same was true in the calculation of silicon clusters by the present workers.[35]

Clearly, the use of real-valued variables and nontraditional genetic operators provides portability of coordinates; thus, there is no need to recast the problem for each new cluster size. In addition, these coordinates in no way restrict the size of the search space. Furthermore, the method seems capable of minimizing problems of chemically interesting size in reasonable CPU times.

The chief purpose of this study is to extend our treatment of clusters to molecular clusters using the same approach. A further goal is to explore systematically the performance of some of the possible operators for real-valued representations in GA calculations. The computational procedures used are described in the next section.

Method

The genetic algorithm approach uses a population of candidate solutions to an optimization problem. Each of these candidate solutions is assigned a fitness related to the value of the objective function evaluated for that candidate. The better the value of the function, the higher the fitness. Candidates are then chosen to exchange genetic material (''to breed'') based on their fitness; the fittest candidates breed most frequently. Offspring are produced from the breeding, their fitness is evaluated, and those sufficient fit are allowed into the genetic pool. Depending on the scheme chosen, less fit individuals may be deleted from the pool to maintain a constant pool size. It is hoped that, after several generations, the population will contain the fittest possible individual, the solution to the optimization problem.

ATOMIC CLUSTERS

For atomic clusters, the candidate solutions are the geometries of the clusters. We use the space-fixed Cartesian coordinates of each atom to describe the geometry. The displacement of the jth atom is given by $x_j = (x_j, y_j, z_j)$. We assume there are N individuals in the genetic pool. We choose the number of individuals in the population to be typically 10 or 20. The ''chromosomes,'' or the individuals in the genetic pool, X_i $(i = 1, N)$, are then the set of SF coordinates of each of the atoms in the n-atom cluster. Thus:

$$X_i = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)$$

The fitness of the individual is obtained from the potential energy of the cluster. The interatomic distances are found using $r_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2$. In this case, we use the traditional Lennard–Jones pairwise additive potential with parameters suitable for argon atoms:

$$V(\mathbf{r}) = 4\varepsilon \sum_{i=1}^{n} \sum_{j>i}^{n} \left(\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right)$$

The values of ε and σ used[38] are 0.012 eV and 3.36 Å, respectively.

Initially, the coordinates are randomly chosen within a box of size L^3 centered at the origin. (In previous work,[34,35] we placed the particles initially in the first octant. We find the present procedure slightly faster.) We take $x_1 = L(\zeta - 0.5)$, etc., where ζ is a freshly generated random number between 0 and 1. We have used $L = \sqrt[3]{6n}\,\sigma$ for the atomic clusters. A conjugate gradient minimization is performed every generation on each individual to place each structure in the vicinity of the nearest minimum. The conjugate gradient procedure is halted if any $r_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \geq L^2$. Also, the magnitude of the gradient in any coordinate is not allowed to exceed 1.0 eV/Å. These measures avoid the possibility of evaporation of the cluster.

MOLECULAR CLUSTERS

We assume that the molecules within the clusters to be treated are rigid. The most economical coordinates to describe the internal coordinates of a rigid body are the three Euler angles.[39] The atomic coordinates of the molecule are conveniently first described in body-fixed (BF) coordinates. For instance, for the H_2O molecule chosen as an example here, one could choose the following:

$$\rho_{\mathrm{O}} = (0,0,0) \qquad \rho_{\mathrm{Ha}} = (r_{\mathrm{OH}} \cdot \cos\gamma, r_{\mathrm{OH}} \cdot \sin\gamma, 0)$$
$$\rho_{\mathrm{Hb}} = (r_{\mathrm{OH}} \cdot \cos\gamma, -r_{\mathrm{OH}} \cdot \sin\gamma, 0)$$

where we have made the particularly simple choice of taking the O atom as the origin, r_{OH} is 0.9572 Å, and γ is 52.26°. A set of Euler angles (θ, ϕ, ψ) is chosen randomly: $0 \leq \alpha \leq 2\pi$, where $\alpha = \theta, \phi, \psi$. Then the molecule is rotated into the SF frame, r_{OHa}, r_{OHb}, by applying a rotation matrix,[39] $R(\theta, \phi, \psi)$, to the vectors ρ_{OHa} and ρ_{OHb}. Finally, the SF coordinates of each atom can be found by adding the displacement of the O atom from the SF origin, R_{O}. Thus, $x_{\mathrm{O}} = R_{\mathrm{O}}$, $x_{\mathrm{OHa}} = R_{\mathrm{O}} + r_{\mathrm{OHa}}$, etc. The potential energy can then be obtained using these SF vectors. The initial R_{O} values are randomly obtained by analogy with the SF coordinates for the atomic clusters. Thus, a (rigid) molecule is completely described by the six coordinates $(R_{\mathrm{O}x}, R_{\mathrm{O}y}, R_{\mathrm{O}z}, \theta, \phi, \psi)$.

The potential used for the H_2O clusters was the TIP3P potentials of Jorgensen et al.[40] Whereas this is acknowledged not to be a very accurate potential, it has been investigated in some detail. In addition, its derivatives are relatively simple.

Each individual's initial $(R_{\mathrm{O}x}, R_{\mathrm{O}y}, R_{\mathrm{O}z})$ coordinates are randomly chosen with a box of size L^3 centered at the origin. We take $R_1 = L(\zeta - 0.5)$, etc., where ζ is a freshly generated random number between 0 and 1. We have used r_e, with $r_e = 2.75$ Å being the dimer OO distance. Each molecule's Euler angles, θ, ϕ, and ψ, are initially generated randomly on the interval $[0, 2\pi]$. A conjugate gradient minimization is performed every generation on each individual to place each structure in the vicinity of the nearest minimum. The same precautions as in the atomic case are taken to avoid evaporation.

The variables for molecular clusters were manipulated in the following fashion. The strings of R and angular coordinates were separated. Thus, for the ith individual, $X_i = (R_{\mathrm{O}x1} \ldots R_{\mathrm{O}zn})$, $Y_i = (\theta_1, \ldots \psi_n)$. Operations were carried out on the X and Y strings separately. For the X list, this was essentially identical to the atomic procedure. For the Y list, the angles θ, ϕ, and ψ are initially defined on the interval $[0, 2\pi]$. Because the trigonometric functions are valid for all values of the angles, we left alone the values produced in the CG (i.e., they were not returned to the $[0, 2\pi]$ interval). After the manipulations, the coordinates for molecule 1 were reassembled by assigning the first three variables of X to be R_{O1}, the first three variables in Y to be (θ, ϕ, ψ) of cluster 1, and so on.

FITNESS FUNCTION AND SELECTION

The participation of an individual in future breeding operations depends upon it fitness. Because the function to be optimized for clusters is the (Born–Oppenheimer) potential energy, the fitness is a function of the potential. There are several schemes for obtaining a fitness function from the value of the potential energy.

Given an individual (i.e., a geometry) X_i, we can calculate the potential energy of the cluster for that geometry, $V_i = V(X_i)$. Given the set $\{V_i: i = 1, n\}$ we can assign the fitness, f_i, of the ith individual. We use the convention that the f_i are normalized to unity. An intermediate quantity, F_i, is evaluated by taking a function of V_i. In most of the results presented here we have used the ''range'' fitness function:

$$F_i = (V_{\max} - V_i)/(V_{\max} - V_{\min}) \qquad i = 1, n$$

The quantities, $V_{\max}$ and $V_{\min}$ are $\max\{V_i\}$ and $\min\{V_i\}$ respectively. The values of f_i are then

found by normalization:

$$f_i = \frac{F_i}{\sum_{i=1}^{N} F_i}$$

The next generation (the ''children'') is formed from the current generation (the ''parents'') as follows. First, the best 20% (i.e., those with the highest fitness) of the individuals in the current generation are passed intact to the next generation (this is known as ''elitism''[14]). The remainder of the population in the next generation is obtained by use of genetic operators on the current generation. These are: (1) inversion (In); (2) geometric mean (Ge); (3) arithmetic mean (Ar); (4) n-point crossover (Nx); and (5) two-point crossover (2x); (6) one-point crossover (1x). These have been partially described by Zeiri.[28] We give the complete details in the Appendix. Of these operators, number 1 transforms one individual into a different individual; numbers 2 and 3 use two parents to ''breed'' one child; numbers 4, 5, and 6 use two parents to produce two children.

All operators are given the same weighting, $w_\alpha = 1/\alpha$, for $\alpha = 1$ through 6, depending on the number of operators we are using in any given run. Following standard Monte Carlo practices,[10] a random number on $[0, 1]$ is generated, and used to determine the operator to be selected. The requisite parents (one or two depending on the operator) are then selectively weighted by their fitness using fresh random numbers.

A typical run contains 10 or 20 individuals in a population. A run is terminated when either the global minimum is found or the potential energy of the fitness structure does not change for five generations.

Results and Discussion

ATOMIC CLUSTERS

Results for a range of $(LJ)_n$ clusters have been presented elsewhere.[34,35] We focus here on the examples of $(LJ)_{13}$ and $(LJ)_{19}$. We have chosen these as examples of a highly symmetric and a rather asymmetric cluster, respectively. For all runs, the CPU time is recorded, as well as the minimum potential energy and the number of generations required to find that minimum. In an earlier study we reported timing data.[34] In this study, we focus on the frequency of location of the global minimum as the chief criterion of the GA's performance. Where appropriate, other data will be mentioned.

Performance of Operators

Zeiri[28] proposed six genetic operators that are appropriate for real-valued genomes. We have tested five of these operators, together with the one-point crossover, on the $(LJ)_{13}$ and $(LJ)_{19}$ clusters, which have global minima that are well known. We have tested each operator individually, and in combination with all other operators. To clarify our findings, we anticipate some of the results. It was seen that the operators fell into three natural groupings, both in form and in performance. These were: (i) inversion (one parent–one child); (ii) averaging (two parents–one child); and (iii) crossovers (two parents–two children).

$(LJ)_{13}$. In Table I we give the number of times the $(LJ)_{13}$ global minimum was reached out of 100 independent attempts for the operators used individually, and in pairs. The diagonal elements are the runs with individual operators. The last column gives the average performance over all six combinations. All calculations reported use the range fitness.

From the diagonal elements, it can be seen that both averaging operators perform very well as the only operator present. The crossover and inversion operators are significantly worse. In combination with any other operator from a different class, however, both crossover and inversion significantly improve their performance. On the other hand, the improvement, when a crossover is paired with another crossover operator, is marginal.

TABLE I.
Number of Times Out of 100 Runs Global Minimum Located for $(LJ)_{13}$ with Combinations of Operators.

Operator							Average
In	50						76
Ar	100	97					98
Ge	98	97	91				96
1x	82	97	95	26			67
2x	67	99	94	70	63		77
Nx	61	100	100	33	66	57	70
	In	Ar	Ge	1x	2x	Nx	

Diagonal elements give performance of operator used alone; off-diagonal elements give performance for two operators used together. ''In'' denotes inversion; ''Ar'' arithmetic mean; ''Ge'' geometric mean; ''1x'' one-point crossover; ''2x'' two-point crossover; ''Nx'' N-point crossover.

Finally, the most robust operators are the averaging operators, performing well with almost any partner.

We now consider $(LJ)_{13}$ minimization with operators taken three at a time—one from each class of operator. The performances are given in Table II. In all cases, the combination of three operators is extremely efficient. In addition, it outperforms the average performance of the same three operators taken two at a time.

We have also carried out all other possible combinations of operators. Rather than enumerate all them, we give average performances over all combinations in which each operator participates. We also include the combination in which it performs worst, and that in which it performs best. These performances are given in Table III. For this particular cluster, it is clear that a judicious combination of operators can easily yield the global minimum on 100% of the attempts. In addition, we note that the mean CPU time (DEC 2100/500) needed to find the global minimum varied from 1.91 to 6.43 seconds per structure, and that most runs took fewer than 20 generations to converge (whether to the global minimum or not). In most cases, the shortest average times and fewest generations were for those cases where the global minimum was located with high frequency.

$(LJ)_{19}$. In Table IV we give the same data for $(LJ)_{19}$ as was shown in Table I. The most striking feature of this table is the drop in the number of ''hits'' in comparison with Table I. This is not surprising; the search space is roughly 50% larger, and the number of local minima has increased dramatically by a ratio of about $\exp(19^2 - 13^2)$. Clearly, the efficiency of the method does not scale with the increase in the number of local minima.

TABLE II.
Performance of Operators Used Three at a Time for $(LJ)_{13}$.

Operators[a]			GM located[b]	Pair average[c]
In	Ar	1x	99	93
In	Ar	2x	100	89
In	Ar	Nx	99	87
In	Ge	1x	98	92
In	Ge	2x	99	86
In	Ge	Nx	99	86

[a] Operators used.
[b] Number of times the global minimum is located in 100 runs.
[c] Average performance of the three operators used in pairs, taken from the data in Table I.

TABLE III.
Summary of Operator's Performance in All Possible Combinations with Other Operators for $(LJ)_{13}$.

Operator	Worst[a]	Average[b]	Best[c]
In	In (50)	91	In, 2x, Ar (100)
Ar	1x, Nx, Ar (93)	98	Nx, Ar (100)
Ge	Ge (91)	97	2x, Nx, Ge (100)
1x	1x (26)	90	1x, 2x, Ar (100)
2x	In, 2x, Nx (60)	91	2x, Nx, Ge (100)
Nx	1x, Nx (33)	90	Nx, Ar (100)

[a] Worst combination in which the operator was used with the number (in parentheses) of times out of 100 in which the global minimum was located.
[b] Average number of times out of 100 the global minimum was located.
[c] Best combination in which the operator participated.

As before, both averaging operators perform very well as the only operator present. The crossover and inversion operators are significantly worse. In combination with any other operator, however, the crossover significantly improves its performance. The inversion improves its performance if combined with an averaging operator. Similarly, the crossovers do well with averaging operators, but not with each other or inversion. Finally, as before, the most robust operators are the averaging operators, performing well with almost any partner.

In Table IV we consider combinations of three operators. As for the $(LJ)_{13}$ case, the use of one operator from each of the three classes is always superior to the average performance of the same operators taken two at a time. However, there is no guarantee that the use of three operators will be better than the best binary combination. Unfortunately, of course, the best combination is generally not known in advance. From examination of

TABLE IV.
Data Same as for Table I, Except $(LJ)_{19}$.

Operator							Average
In	7						26
Ar	68	57					64
Ge	57	67	54				61
1x	13	55	62	0			24
2x	4	55	49	13	5		22
Nx	4	81	74	1	5	0	28
	In	Ar	Ge	1x	2x	Nx	

Table V, it appears that the two-point crossover tends to be less effective than either the one-point or the *n*-point.

Table VI is analogous to Table III. The best combinations of operators clearly always include an averaging operator; the worst, a crossover—most frequently the two-point. From what we have seen above, however, it is a good strategy to include at least one crossover operation. It would appear that either the one-point or the *n*-point is, in general, to be preferred over the two-point.

For the $(LJ)_{19}$ case, the mean CPU times range from 4 to 30 seconds per structure, and typically 10 to 20 generations are required for convergence.

Slice Operator

Deaven and coworkers[29,30] have suggested another genetic operator which we have not tested in previous works. In general, this operation is carried out as follows. The cluster is cut by a plane which contains the centroid of the cluster, yielding two subclusters. Subclusters arbitrarily considered to be ''above'' the plane are then added to subclusters ''below'' the plane. The resulting cluster (after suitable adjustments to ensure that the number of atoms is correct) is relaxed using a CG minimization. Sufficiently fit offspring clusters are accepted into the population. Deaven and coworkers[29,30] used all possible above–below combinations in their work. To compare their operator with those described above, we use a slightly different procedure here. As with the crossover operators described above, two candidate clusters are selected based on their fitness, and the offspring are the two above–below combinations of the subclusters. More details are given in the Appendixes.

The results using this operator alone are as follows. For the $(LJ)_{13}$ cluster, the global minimum was located in 66 out of 100 independent attempts (mean CPU time 10.35 seconds; mean number of generations 29). By comparison with Figure 1, we see that this is comparable to the better crossover operators for this cluster size, but less efficient than the averaging operators.

TABLE V.
Data Same as for Table II, Except $(LJ)_{19}$.

Operators			GM located	Pair average
In	Ar	1x	75	45
In	Ar	2x	55	42
In	Ar	Nx	79	51
In	Ge	1x	77	44
In	Ge	2x	62	36
In	Ge	Nx	72	45

TABLE VI.
Data Same as for Table III, Except $(LJ)_{19}$.

Operator	Worst	Average	Best
In	In, 2x (4)	51	In, Nx, Ar (79)
Ar	2x, Ar, Ge (44)	61	Nx, Ar (81)
Ge	2x, Ar, Ge (44)	62	In, 1x, Ge (77)
1x	1x (0)	51	In, 1x, Ge (77)
2x	In, 2x, Nx (3)	46	In, 1x, 2x, Nx, Ge (70)
Nx	Nx (0)	51	Nx, Ar (81)

For the $(LJ)_{19}$ case, the global minimum is found in only 7 out of 100 runs (mean CPU time 18.84 seconds; mean number of generations 89). Again, this is comparable to inversion and the two-point crossover for this cluster size, but inferior to the averaging operators.

Other Fitness Functions

To explore the possible effects of the choice of fitness parameter we have carried out calculations using an exponential fitness scheme. In this case, the intermediate value, F_i, is calculated as:

$$F_i = \exp[-\alpha V_i], \qquad i = 1, n$$

and the f_i are normalized as stated previously for the range fitness function. We have judiciously chosen the two values of α we investigate here, $\alpha = 2$ and $\alpha = 5$, to illustrate the possible consequences of using an exponential scheme. We report results for a system of 13 LJ atoms. The minima in this system have potential values ranging from 0 to -0.545 eV.

In Figure 1, we show the fitness functions over this energy interval for the two exponential schemes, and the range scheme, suitably normalized. It can be seen that the range preferentially selects lower values of the potential. The exp-2 is rather less selective, whereas the exp-5 is strongly weighted toward low potential values.

We show results for $(LJ)_{13}$ using the individual operators and the three fitness schemes in Table VII. In each case, the best performance is underlined. It is clear that there is little to choose between the three schemes. However, exp-5, which weights the fittest structure in the population very heavily, performs most poorly. Therefore, it does not appear to be productive in this case to be

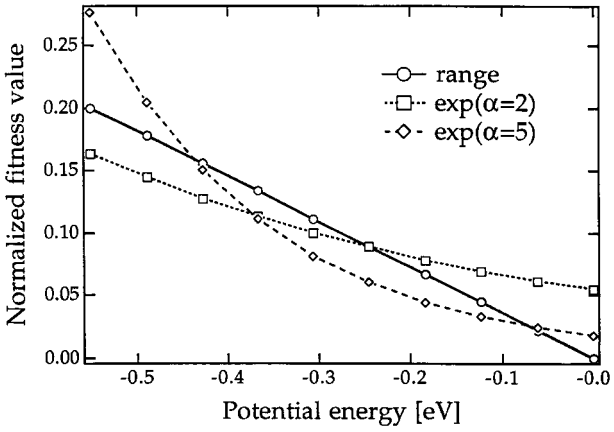

FIGURE 1. Fitness as a function of energy for three different fitness functions (see text). The minimum energy is that of $(LJ)_{13}$.

TABLE VII.
Performance of Individual Operators for $(LJ)_{13}$ for Three Different Fitness Functions (See Text). Data are Number of Times Global Minimum is Located Out of 100 Runs.

	Range	exp(α = 2)	exp(α = 5)
In	50	56	52
Ar	97	100	92
Ge	91	97	90
1x	26	24	24
2x	63	69	66
Nx	57	50	50
Average	64	66	62

highly selective; it appears that maintaining a diverse population is advantageous.

MOLECULAR CLUSTERS

By analogy with our earlier work on atomic clusters,[34,35] we have explored the feasibility of locating the global minimum for $(H_2O)_n$ using all the genetic operators. Statistics for the runs with $n = 2$–8 are given in Table VIII. It can be seen that most of the smaller water clusters are trivial to minimize in the sense that almost any random initial geometry descends to the global minimum. For $n = 7$ and 8, the task is more challenging. It can be seen that the computational effort required to locate the global minimum rises dramatically as the size of the cluster grows. It is possible that we would be able to locate the minimum if we allowed the population a longer evolution period. However, this would seem to be needless waste of resources.

For this reason, we treat the clusters with $n = 9$ through 13 in a slightly different manner. In these cases, we start ten individual populations, and evolve each in the usual way until the convergence criteria are met. We then construct a new population from the fittest of each of the initial populations, and evolve this new population. In all cases, there was some improvement in the potential energy. The geometries of $(H_2O)_n$ for $n = 2$ through 13 are shown in Figures 2 and 3. We are in agreement with published geometries and energies[12] for $n = 8$. The larger clusters all show the ''fused

TABLE VIII.
Summary of Performance for $(H_2O)_n$, (n = 2–8).

n	$-V_{max}$[a]	$-\bar{V}$[b]	$-V_{min}$[c]	$\Gamma_{max}, \bar{\Gamma}, \Gamma_{min}$[d]	#V_{min}/100[e]	$\bar{t}$ [sec][f]
2	6.543	6.543	6.543	0, 0.00, 0	100	1.265
3	17.448	17.448	17.448	0, 0.00, 0	100	6.362
4	29.306	29.306	29.306	0, 0.00, 0	100	21.377
5	38.771	38.771	38.771	1, 0.06, 0	100	50.291
6	47.811	47.811	47.811	3, 0.24, 0	100	88.223
7	57.425	57.634	57.944	24, 10.70, 0	39	1087.740
8	67.594	68.900	70.681	12, 17.82, 0	10	2276.005

[a] Lowest local minimum potential (presumed to be global minimum).
[b] Mean value of the potential in the population.
[c] Highest local minimum found.
[d] Maximum, mean, and minimum number of generations before convergence criteria were met.
[e] Number of times out of 100 that the global minimum was located.
[f] Mean CPU time for minimizing each structure.

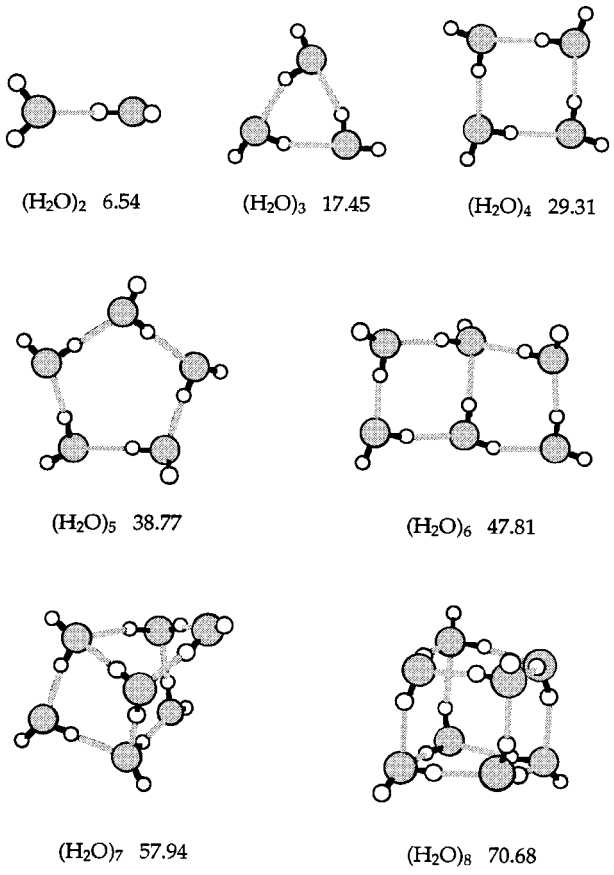

FIGURE 2. Geometry of lowest potential energy structure found for $(H_2O)_n$, $n = 2-8$.

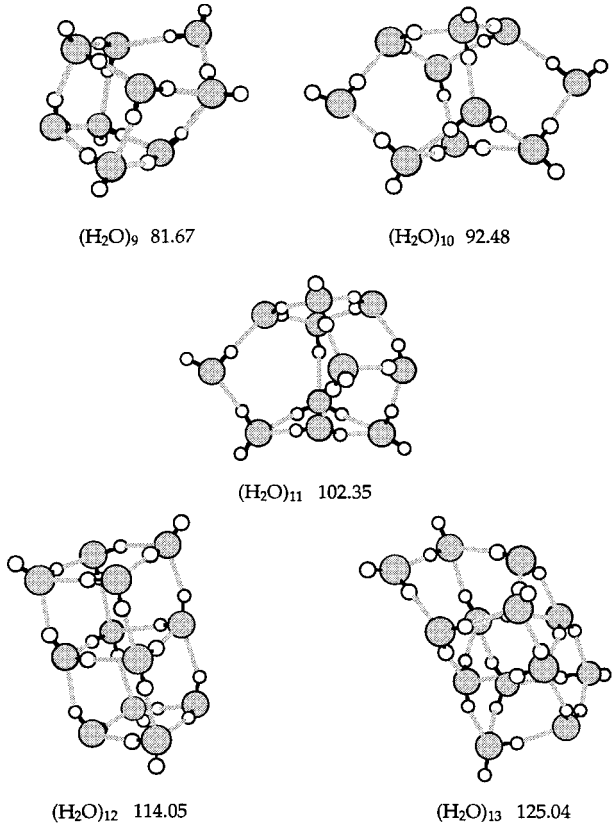

FIGURE 3. As for Figure 2, except $n = 9-13$.

cube'' seen in studies[41] of water clusters with $n = 8$, 12, 16, and 20.

Performance of Operators

The particular interest in our study of molecular clusters lies in the evaluation of the operators on the angular part of the problem in particular. This has received considerably less attention than the treatment of the center-of-mass coordinates. To isolate this performance, we carry out calculations on a $(H_2O)_8$ cluster with the eight O atoms frozen at the geometry of the global minimum. The operators then act only on the Euler angles. Table IX is similar to Tables I and IV, and documents the success of the operators individually and in pairs.

For the Euler angles, the best operators, both individually and in combination, are the two-point crossover and inversion. The remaining operators are fair. It is unclear why these results are in such contrast to those for the Cartesian coordinates. We will investigate this in future work.

We have also carried out runs with all possible combinations of operators for this system. Our qualitative findings remain unaltered; therefore, we refrain from showing these extra details.

Conclusions

We have shown that we can extend earlier work[34,35] using a modified genetic algorithm approach in space-fixed real variables to rigid molec-

TABLE IX.
Data Same as for Table I, Except for $(H_2O)_8$, with O Atoms Fixed (See Text).

Operator							Average
In	61						65
Ar	59	46					58
Ge	69	54	44				58
1x	63	60	52	34			54
2x	70	67	64	75	73		70
Nx	66	60	62	42	70	44	57
	In	Ar	Ge	1x	2x	Nx	

ular clusters. This was done by using a Cartesian space-fixed reference vector with each molecule, and describing the internal coordinates of each molecule by the three Euler angles. The populations of Cartesian and angular coordinates were acted upon separately by the genetic operators. Using this method we were able to locate global minima for $(H_2O)_n$, $n = 2-13$, using relatively little CPU time.

We have also tested the efficiency of the proposed genetic operators singly and in combination with other operators. For atomic clusters, averaging operators are clearly the most efficient if used individually, but may improve their performance in judicious combination. Because the correct combination is not known *a priori* for any given problem, we recommend the use of a mix of operators: one averaging, one crossover, and inversion.

For atomic clusters, we have compared these operators with a ''slice'' operator similar to that employed by Deaven and coworkers.[29,30] We found this operator to be comparable in efficiency to crossover operators, but not as good as averaging operators. It is not clear whether there is an analogous operator for angular coordinates. We, therefore, made no attempt to implement this approach for molecular clusters.

We also investigated the possibility of using different fitness functions for atomic clusters. It appears that trying to bias the selection toward the very fittest individuals is counterproductive; the genetic algorithm appears to work best when there is reasonable diversity in the population.

Clearly, there is considerable scope in the genetic algorithm field for the development of new ''custom'' operators tailored to the problem at hand. We are currently investigating such operators.

Appendix A

We give the details needed to fully understand the genetic algorithm operators used. We denote the ith geometry of the n-atom cluster as $X_i = (x_1, \ldots, x_n)$, where $x_k = (x_k, y_k, z_k)$ is the displacement of the kth atom. While the distinction between x, y, and z coordinates is important for evaluating the potential, the operators act simply on a string of reals. To emphasize this we relabel this string of reals as $X_i = (c_1, \ldots, c_{3n})$. We use c_k to denote $c_k(i)$ if there is no ambiguity. Below we summarize the action of each of the operators. In all the expressions that follow k runs from 1 to $3n$. In the notation $[c_k(i), c_k(j)] = [c_k(j), c_k(i)]$ simultaneous substitution is implied, with the updated generation on the left-hand side and the current generation on the right-hand side of the assignment. We also include an example of each operator's behavior.

Inversion:

$c_k = c_{q+r-k}$, $r \le k \le q$, r, q flat on $[1, 3n]$. A single parent, $[\alpha_1, \ldots, \alpha_{k-1}, \alpha_k, \alpha_{k+1}, \ldots, \alpha_{3n}]$, is required for inversion. For instance, if $r = k - 2$ and $q = k + 1$, the resulting child is $[\alpha_1, \ldots, \alpha_{k+1}, \alpha_k, \alpha_{k-1}, \ldots, \alpha_{3n}]$.

One-point crossover:

$[c_k(i), c_k(j)] = [c_k(j), c_k(i)]$, $s < k \le 3n$, s flat on $[1, 3n]$. Two parents, $[\alpha_1, \ldots, \alpha_{k-1}, \alpha_k, \alpha_{k+1}, \ldots, \alpha_{3n}]$ and $[\beta_1, \ldots, \beta_{k-1}, \beta_k, \beta_{k+1}, \ldots, \beta_{3n}]$, produce two children. For example, $[\beta_1, \ldots, \beta_{k-1}, \alpha_k, \alpha_{k+1}, \ldots, \alpha_{3n}]$ and $[\alpha_1, \ldots, \alpha_{k-1}, \beta_k, \beta_{k+1}, \ldots, \beta_{3n}]$ result if $s = k-1$.

Two-point crossover:

$[c_k(i), c_k(j)] = [S_{s+k}(ij), S_{s+k+3n}(ij)]$, s flat on $[1, 3n]$. $S(ij) = (c_1(i), \ldots, c_{3n}(i), c_1(j), \ldots, c_{3n}j))$ and it is understood that $s + k + 3n$ is modulo $6n$. The two parents, $[\alpha_1, \ldots, \alpha_{k-1}, \alpha_k, \alpha_{k+1}, \ldots, \alpha_{3n}]$ and $[\beta_1, \ldots, \beta_{k-1}, \beta_k, \beta_{k+1}, \ldots, \beta_{3n}]$, yield two children, $[\alpha_{k-1}, \alpha_k, \alpha_{k+1}, \ldots, \alpha_{3n}, \beta_1, \ldots, \beta_{k+2}]$ and $[\beta_{k-1}, \beta_k, \beta_{k+1}, \ldots, \beta_{3n}, \alpha_1, \ldots, \alpha_{k-2}]$ if, for example, $s = k - 2$.

N-point crossover:

$[c_k(i), c_k(j)] = [c_k(j), c_k(i)]$, if $\zeta > 0.5$. $[c_k(i), c_k(j)] = [c_k(i), c_k(j)]$, if $\zeta \le 0.5$. Two parents, $[\alpha_1, \ldots, \alpha_{k-1}, \alpha_k, \alpha_{k+1}, \ldots, \alpha_{3n}]$ and $[\beta_1, \ldots, \beta_{k-1}, \beta_k, \beta_{k+1}, \ldots, \beta_{3n}]$, produce two children. For example, $[\beta_1, \ldots, \alpha_{k-1}, \beta_k, \beta_{k+1}, \ldots, \alpha_{3n}]$ and $[\alpha_1, \ldots, \beta_{k-1}, \alpha_k, \alpha_{k+1}, \ldots, \beta_{3n}]$ may result, depending on the $3n$ ''fresh'' random numbers, ζ.

Arithmetic mean:

$c_k(i) = 0.5(c_k(i) + c_k(j))$ Two parents, $[\alpha_1, \ldots, \alpha_n]$ and $[\beta_1, \ldots, \beta_{3n}]$ produce one child, $[0.5(\alpha_1 + \beta_1), \ldots, 0.5(\alpha_{3n} + \beta_{3n})]$.

Geometric mean:

$c_k(i) = (\mathrm{abs}(c_k(i) \cdot c_k(j)))^{1/2}$ Two parents, $[\alpha_1, \ldots, \alpha_{3n}]$ and $[\beta_1, \ldots, \beta_{3n}]$, yield one child, $[\{\mathrm{abs}(\alpha_1 \cdot \beta_1)\}^{1/2}, \ldots, \{\mathrm{abs}(\alpha_{3n} \cdot \beta_{3n})\}^{1/2}]$.

Slice operation:

Two parents, $[\alpha_1, \ldots, \alpha_{k-1}, \alpha_k, \alpha_{k+1}, \ldots, \alpha_{3n}]$ and $[\beta_1, \ldots, \beta_{k-1}, \beta_k, \beta_{k+1}, \ldots, \beta_{3n}]$, produce two children. For example, $[\beta_1, \ldots, \alpha_{k-1},$

$\beta_k, \beta_{k+1}, \ldots, \alpha_{3n}]$ and $[\alpha_1, \ldots, \beta_{k-1}, \alpha_k, \alpha_{k+1}, \ldots, \beta_{3n}]$ may result, depending on the randomly chosen plane.

Appendix B

The slice operation is carried out as follows. Two parents are selected as usual, based upon their fitness values. A plane containing the space-fixed origin is generated by randomly choosing the two spherical polar angles, θ and ϕ. The cosine of angle θ is chosen flat on -1 to 1, whereas ϕ is chosen flat on 0 to 2π. These two angles generate a random vector to the surface of a sphere of unit radius. A plane is then generated perpendicular to this vector.

It is decided whether an atom is ''above'' or ''below'' the plane by the sign of the dot product of the vector from the atom to the origin, with the plane's normal vector at the origin. We ensure that each cluster contains the correct number of atoms by adding atoms from the opposite ''side'' of the plane, if necessary.

Appendix C

DUPLICATION

There are several ways in which the above operators can allow duplication of individuals within a population. To be duplicates here means that two structures have not only the same potential energy (degenerate), but have the same coordinates as well. The population may become overly weighted with duplicates of the lowest energy structure, because it is chosen most often to be a parent (has the highest fitness). A second reason why duplication is undesirable, particularly within a population as small as the one used here, is because duplicate parents can exchange information resulting in a child cluster containing two atoms with identical coordinates. We avoid most duplications by preventing certain choices during some of the breeding schemes. These are:

Inversion:	$k \neq 3n$.
One-point crossover:	$k \neq 3n$ and $i \neq j$.
Two-point crossover:	$k \neq 3n$ and $i \neq j$.
N-point crossover:	$i \neq j$ and we ensure that at least one switch occurs.
Arithmetic mean:	$i \neq j$.
Geometric mean:	$i \neq j$.

However, duplication may still occur through more complicated, but rare, manipulations spanning more than one generation. We have scanned some runs for structures with the same energy within any one generation. We determined if they were duplicates, and not merely degenerate, by simply subtracting corresponding coordinates. A result of zero for each of the $3n$ pairs indicates duplication. We found, after the above restrictions were implemented, that fewer than 0.1% of generations (10 individuals in population, $n = 13$) contained duplicates. As mentioned previously, duplication may result in a child cluster that contains two or more atoms with identical coordinates, causing divide-by-zero errors upon calculation of cluster potential. We prevent these errors by artificially setting any $r_{ij}^2 < 8.0$ Å^2 to 8.0 Å^2 (for LJ clusters) during the calculation of the potential, without actually altering the coordinates themselves. Similar appropriate precautions are taken with the molecular clusters. This ensures that the offending structure, unless it is significantly improved during local minimization, receives an extremely low fitness and is subsequently eliminated from the population.

Appendix D

A preliminary version of the code used here is available from the authors (hrm@christa.unh.edu).

References

1. A. R. Leach, *Rev. Comput. Chem.*, **2**, 1 (1991).
2. H. Dodziuk, *Modern Conformational Analysis*, VCH, New York, 1995.
3. P. Comba and T. W. Hambley, *Molecular Modeling*, VCH, New York, 1995.
4. M. R. Hoare, *Adv. Chem. Phys.*, **40**, 49 (1979).
5. M. R. Hoare and P. Pal, *Adv. Phys.*, **20**, 161 (1971).
6. J. A. Northby, *J. Chem. Phys.*, **87**, 6166 (1987).
7. C. D. Maranas and C. A. Floudas, *J. Chem. Phys.*, **97**, 7667 (1992).
8. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes*, 2nd ed., Cambridge University Press, New York, 1992.
9. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Science*, **220**, 671 (1983).
10. M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods*, John Wiley & Sons, New York, 1986.
11. D. D. Frantz, D. L. Freeman, and J. D. Doll, *J. Chem. Phys.*, **93**, 2769 (1990).

12. C. J. Tsai and K. D. Jordan, *J. Chem. Phys.*, **99**, 6957 (1993).
13. P. Amara, D. Hsu, and J. E. Straub, *J. Phys. Chem.*, **97**, 6715 (1993).
14. M. Sylvain and R. L. Somorjai, *J. Phys. Chem.*, **95**, 4147 (1991).
15. A. B. Finnila, M. A. Gomez, C. Sebenik, C. Stenson, and J. D. Doll, *Chem. Phys. Lett.*, **219**, 343 (1994).
16. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
17. L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand, Reinhold, New York, 1991.
18. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, New York, 1994.
19. D. E. Goldberg, *Complex Syst.*, **5**, 139 (1991).
20. R. S. Judson, M. E. Colvin, J. C. Meza, A. Huffer, and D. Gutierrez, *Int. J. Quant. Chem.*, **44**, 277 (1992).
21. R. S. Judson, *J. Phys. Chem.*, **96**, 10102 (1992).
22. D. B. McGarrah and R. S. Judson, *J. Comput. Chem.*, **14**, 1385 (1993).
23. P. Tuffery, C. Etchebest, S. Hazout, and R. Lavery, *J. Biomol. Struct. Dynam.*, **8**, 1267 (1991).
24. P. Tuffery, C. Etchebest, S. Hazout, and R. Lavery, *J. Comput. Chem.*, **14**, 790 (1993).
25. W. Linert, P. Margl, and I. Lukovits, *Comput. Chem.*, **16**, 61 (1992).
26. R. S. Judson, E. P. Jaeger, and A. M. Treasurywala, *J. Mol. Struct.* (*Theochem*), **308**, 191 (1994).
27. B. Hartke, *J. Phys. Chem.*, **97**, 9973 (1993).
28. Y. Zeiri, *Phys. Rev.*, **E51**, R2769 (1995).
29. D. M. Deaven and K. M. Ho, *Phys. Rev. Lett.*, **75**, 288 (1995).
30. D. M. Deaven and N. Tit, J. R. Morris, and K. M. Ho, *Chem. Phys. Lett.*, **256**, 195 (1996).
31. B. Hartke, *Chem. Phys. Lett.*, **240**, 560 (1995).
32. J. Mestres and G. Scuseria, *J. Comput. Chem.*, **16**, 729 (1995).
33. S. K. Gregurick, M. H. Alexander, and B. Hartke, *J. Chem. Phys.*, **104**, 2684 (1996).
34. J. A. Niesse and H. R. Mayne, *J. Chem. Phys.*, **105**, 4700 (1996).
35. J. A. Niesse and H. R. Mayne, *Chem. Phys. Lett.*, **261**, 576 (1996).
36. Y. Xiao and D. E. Williams, *Chem. Phys. Lett.*, **215**, 17 (1993).
37. J. N. Beauregard and H. R. Mayne, *Surf. Sci. Lett.*, **280**, L253 (1993).
38. G.-Q. Xu, R. J. Holland, S. L. Bernasek, and J. C. Tully, *J. Chem. Phys.*, **90**, 3831 (1989).
39. H. Goldstein, *Classical Mechanics*, Addison-Wesley, Cambridge, MA, 1953.
40. W. L. Jorgensen, J. Chandresekhar, J. D. Madura, R. W. Impey, and M. L. Klein, *J. Chem. Phys.*, **79**, 926 (1983).
41. C. J. Tsai and K. D. Jordan, *J. Phys. Chem.*, **97**, 5208 (1993).